



User's Manual

IK 342 VMEbus Counter Card

Contents

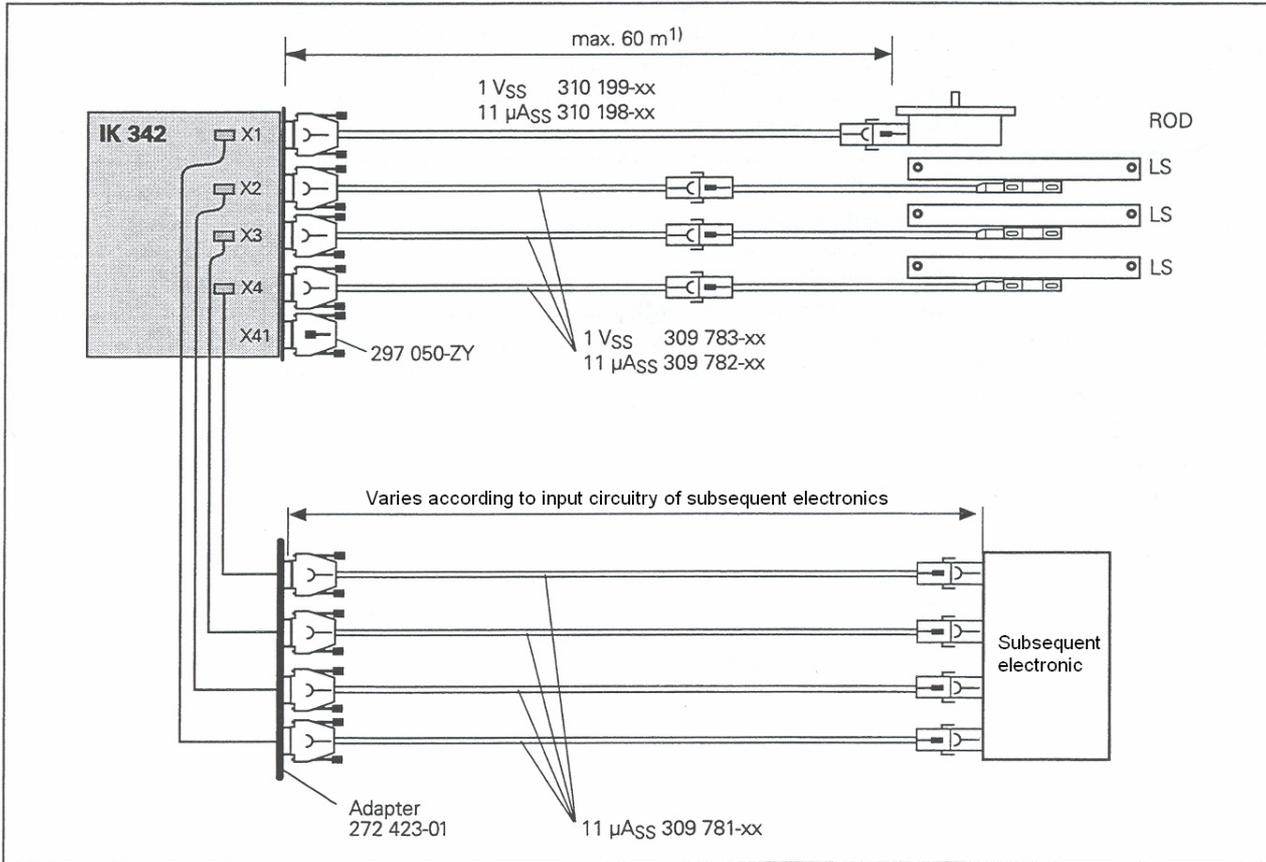
Contents	2
1 Items supplied	4
1.1 Accessories	4
2 Important Points	5
3 Technical Description of the IK 342	6
3.1 Access time to measured values	7
4 Hardware	8
4.1 VMEbus interface	8
4.2 Encoder inputs IK 342	8
Specification of the signals 1 V_{PP}	8
Specification of the signals 11 μA_{PP}	8
Connectors X1 to X4 for encoders	9
4.3 Encoder outputs	10
4.4 Compensating the encoder signals	10
4.5 External functions	12
5 Addressing	13
5.1 Allocation of the address space	13
5.2 Switches and jumpers	14
5.3 Interrupts	15
6 Latching a Position Value	16
6.1 Overview	16
6.2 Latching with software	16
6.3 Latching with hardware via X41	16
6.4 latching with several cards via external cascading	17
6.5 latching with reference mark	17
6.6 Latching with timer	18
7 Registers	18
7.1 BA + \$0000: ID register (read only access)	18
7.2 Registers of the counter ICs	18
Register overview	19
\$28 to \$3C: Data registers for the counters	19
\$24: Initialization register 1 (write access)	20
\$24: Initialization register 2 (write access)	21
\$20: Control register 1 (write access)	22
\$20: Status register 1 (read access)	22
\$20: Status register 2 (read access)	22
\$1C: Reference marks register (write access)	23
\$1C: Amplitude value register {read access}	23
\$18: Register for enabling measured value interrogation {write access}	24
\$18: Register for axis cascading (write access)	24
\$10: Offset register for the 0° signal (write access)	25
\$10: Amplitude for the 0° signal (read access)	25
\$0C: Offset register for the 90° signal (write access)	26
\$0C: Amplitude for the 90° signal (read access)	26
\$08: Timer register (write access)	27
\$04: Control register 2 (write access)	28
\$04: Status register 3 (read access)	28
\$04: Identification register (read access)	29
\$00: Control register 3 (write access)	29
\$00: Status register 4 (read access)	29

7.3 BA + \$180: Registers for configuring the latch logic	30
\$00 to \$06: Configuration register for the external inputs EI to E4 (read and write access).....	31
\$08: Configuration register for the input levels EI to E4 and configuration of encoder input X1, X2 (read and write access)	31
\$0C Configuration register for encoder inputs X2, X3, X4 (read and write access)	32
\$0C: Register for display of the latch source (read access).....	32
\$0C: Register for display of the latch source (write access)	33
\$0E: Configuration register for the I ² C bus, interrupt enable/disable, excess of input signal level (read access).....	33
\$0E: Configuration register for the I ² C bus, interrupt enable/disable, excess of input signal level (write access)	33
8. Programming	34
8.1 Basic functions	34
8.1.1 The header file IK342_0.H	35
8.1.2 The functions in IK342_0.C	36
8.1.3 The header file SAMPLE.H	40
8.1.4 Program example SAMPLE32.C	40
8.1.5 Program example SAMPLE48.C	42
8.2 Functions for RAM model	43
9 Specifications of the IK 342	45
10 Block Diagram of the Latch Paths in the Counter ICs	46

1 Items Supplied

VMEbus counter card IK 342 with encoder inputs for sinusoidal signals ($1 V_{PP}$ or $11 \mu A_{PP}$ switchable), programming examples, driver software end User's Manual

1.1 Accessories



1) Cables up to 150 m are possible if it is guaranteed that the encoder will be supplied by 5 V from an external power source.

310 199-xx 15/12-pin	Adapter cable with connector for HEIDENHAIN encoders ($1V_{PP}$) with flange socket; standard length.0.5 m
310 198-xx 15/9-pin	Adapter cable with connector for HEIDENHAIN encoders ($11 \mu A_{PP}$) with flange socket; standard length.0.5 m
309 387-xx 15/12-pin	Adapter cable with coupling for HEIDENHAIN encoders ($1V_{PP}$) standard length 0.5 m
309 382-xx 15/9-pin	Adapter cable with coupling for HEIDENHAIN encoders ($11 \mu A_{PP}$) standard length 0.5 m
297 050-ZY	Connector for external functions at connection X41
272 423-01	Adapter for encoder outputs (sinusoidal current signals)
309 781 xx	Connection cable from encoder output to another display or contra!

2 Important Points



Danger to internal components!

When handling components that can be damaged by **electrostatic discharge (ESD)**, observe the safety recommendations in EN 100015. Use only antistatic packaging material. Be sure that the work station and the technician are properly grounded during installation.

Some points on the terms used:

- Numbers in hexadecimal notation are identified by \$, e.g. \$FF.
- Inverted signals have a minus sign before their name, e.g. -PULSE1.
- The term "**to latch**" means that the counter value is stored in the data register. This count value must then be **interrogated**. i.e. read by the software and stored in the computer or displayed on the screen.
- Please refer to the technical literature on the VMEbus for the significance of the VMEbus Signals and related terms.

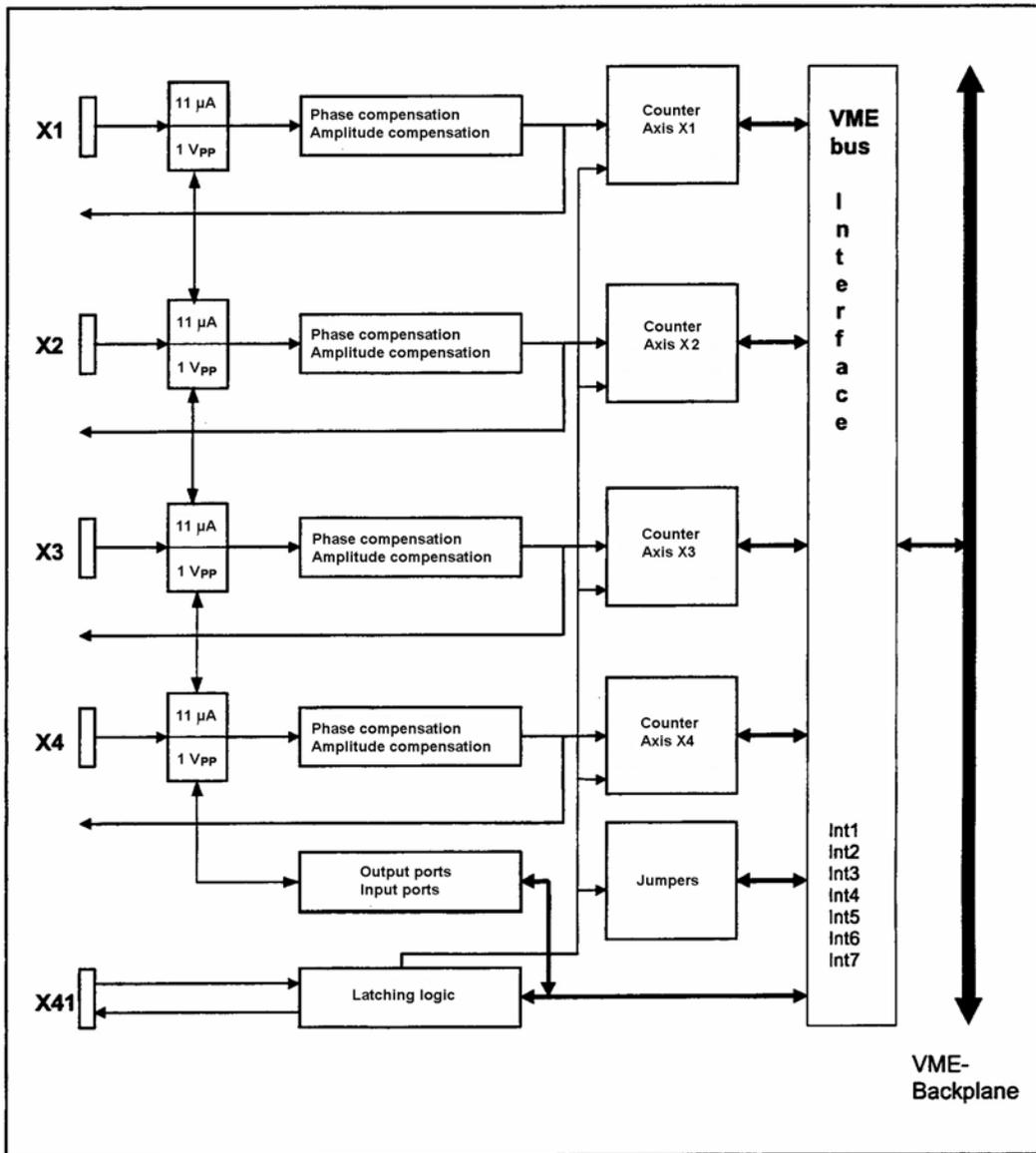
3 Technical Description of the IK 342

The IK 342 is a counter card for measurement of distance and angles with the help of VMEbus computers. You can connect your HEIDENHAIN encoders with sinusoidal voltage and current signals to the IK 342. The card is then inserted directly into a free slot in the computer.

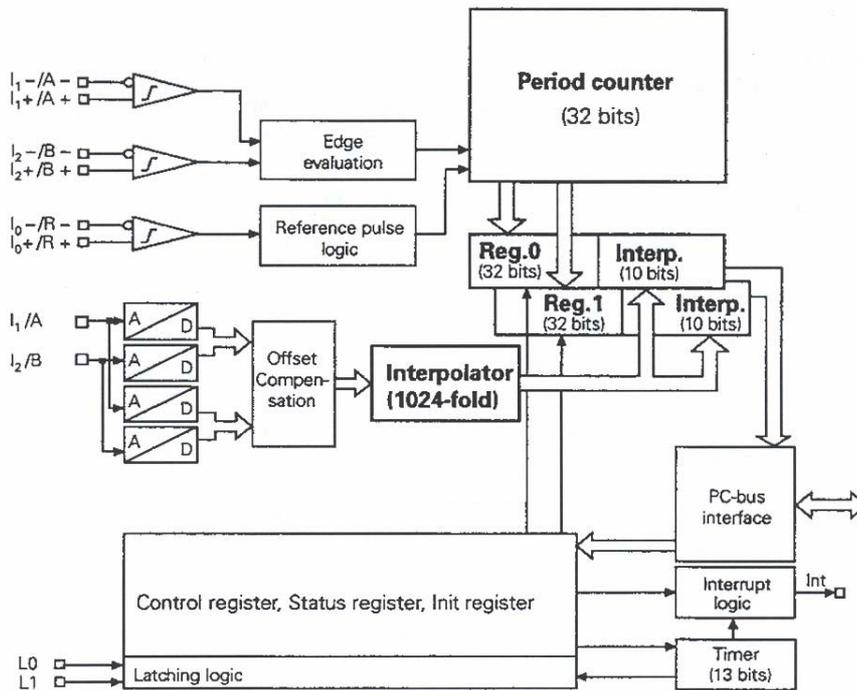
You can interrogate the positions of the four encoders with external polling inputs or with software, and then the positions can be further processed in the computer.

The IK 342 is ideal for applications for which high resolution of encoder signals and high-speed data acquisition are necessary.

Block diagram of the IK 342:



Block diagram of the counter IC:



The interpolation electronics in the IK 342 subdivides the signal period of the input signal 1024-fold. The 10-bit interpolation value together with the 32-bit value of the period counter form the 42-bit measurement value. The IK 342 stores the measurement values in 48-bit data registers, whereby the top bits are expanded with the correct sign in two's complement representation.

You can latch the measured values either via external outputs, via software, via timer or by traversing the reference marks, and then transfer the values to the computer on the VMEbus.

You can compensate phase and amplitude of the sinusoidal encoder signals via electronic potentiometer using software, and the offset via data registers in the counter IC.

3.1 Access time to measured values

The time necessary for accessing a measured value is $< 25 \mu\text{s}$.

4 Hardware

4.1 VMEbus interface

Specification:	ANSI/IEEE STD 1014-1987, IEC 821 and 297
Size:	Double height board (170 mm x 261 mm), 1 slot
Slave:	A16, D16, D08(EO). D08(O) interrupter
Interrupt lines:	7
Current consumption (max):	+ 12 V: 25 mA - 12 V: 25 mA +5 V: 350 mA (without encoders)
Power consumption (max):	2.5 W without encoders

4.2 Encoder inputs IK 342

You can connect HEIDENHAIN linear or angle encoders with sinusoidal voltage signals A and B or with sinusoidal current signals I1 and I2 to the IK 342. The encoder inputs can be switched via software and the input frequency is programmable (view registers BA + \$180 + \$08 and BA + \$180 + \$0A).

Specification of the encoder inputs 1 V_{PP}

Signal amplitudes: A, B (0°, 90°) R (reference mark)	0,6 V _{PP} to 1,2 V _{PP} 0,2 V to 0,85 V usable component
Signal levels for error message A, B	≤ 0,22 V _{PP}
Maximum input frequency	300 kHz
Cable length ¹⁾	Max. 60 m (operational voltage 5,0 V)

1) Cables up to 150 m are possible if it is guaranteed that the encoder will be supplied by 5 V from an external power source. In this case the input frequency is reduced to max. 250 kHz.

Specification of the encoder inputs 11 μA_{PP}

Signal amplitudes: I ₁ , I ₂ (0°, 90°) I ₀ (reference mark)	7 μA _{PP} to 16 μA _{PP} 3,5 μA _{PP} to 8 μA _{PP} usable component
Signal levels for error message I ₁ , I ₂	≤ 2,5 μA _{PP} ≥ 25,6 μA _{PP}
Maximum input frequency	switchable 33 kHz / 175 kHz
Cable length	Max. 30 m (operational voltage 5,0 V)

Connections X 1 to X4 for encoders
D-sub connector with male contact (15-pin)

Connection no.	Allocation 1 V _{PP}	Allocation 11 μA _{PP}
1	+5 V (U _P)	+5 V (U _P)
2	0 V(U _N)	0 V(U _N)
3	A+	I ₁ +
4	A-	I ₁ -
5	0 V	0 V
6	B+	I ₂ +
7	B-	I ₂ --
B	0 V	0 V
9	+5 V	+5 V
10	R+	I ₀ +
11	0 V	0 V
12	R-	I ₀ -
13	0 V	0 V
14	Not assigned	Not assigned
15	Not assigned	Not assigned
Housing	External! shield	External! shield

4.3 Encoder outputs

The IK 342 also outputs the encoder signals of connections XI to X4 in the form of sinusoidal current signals via four 10-pin AMP connectors (XI1, XI2, XI3 and XI4) on the PCB ($11 \mu A_{PP}$)¹⁾ These signals can be sent out to 9-pin D-sub connectors via an additional assembly (Id.-Nr. 272 423-01). Adapter cables (Id.-Nr. 309 781-..) for connection to HEIDENHAIN position display units or interpolation electronics can be supplied (see "1.1 Accessories").

Encoder outputs

D-sub connector with male contact {9-pin}

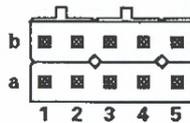
Connection no.	Allocation
1	$I_1 -$
2	OV(UN)
3	$I_2 -$
4	Not connected
5	$I_0 -$
6	$I_1 +$
7	Not connected
8	$I_2 +$
9	$I_0 +$
Housing	External shield

PCB connector for encoder outputs

AMP with male contact (10-pin)

Connection no. ¹⁾	Signal
1a	Not connected
1b	Not connected
2a	Not connected
2b	0V (U_N)
3a	$I_0 -$
3b	$I_0 +$
4a	$I_2 -$
4b	$I_2 +$
5a	$I_1 -$
5b	$I_1 +$

1) The side with the locking pins is 'b'.
Connections 1a and 1b are on the side with the notch.



4.4 Compensating the encoder signals

The sinusoidal encoder signals should be compensated in cases where it is required that the measured values are very accurate. The signals can be compensated using software and the following values can be compensated:

- Phase and amplitude via electronic potentiometer
- Symmetry (offset) in the counter ICs with offset registers

The potentiometer is controlled by 12C-bus, which in turn is controlled by a register of the configuration logic and/or latch logic (address \$0E, bit0, bit1, bit2). As producing the control sequences requires a lot of time and effort, we recommend that you use the program POTIS.EXE

or, if this will not run on your computer, that you use the functions and procedures in "BORLAND C++" from the files IIC.CPP, POT1_1.CPP and POTIS.CPP, as an orientation help for your own functions.

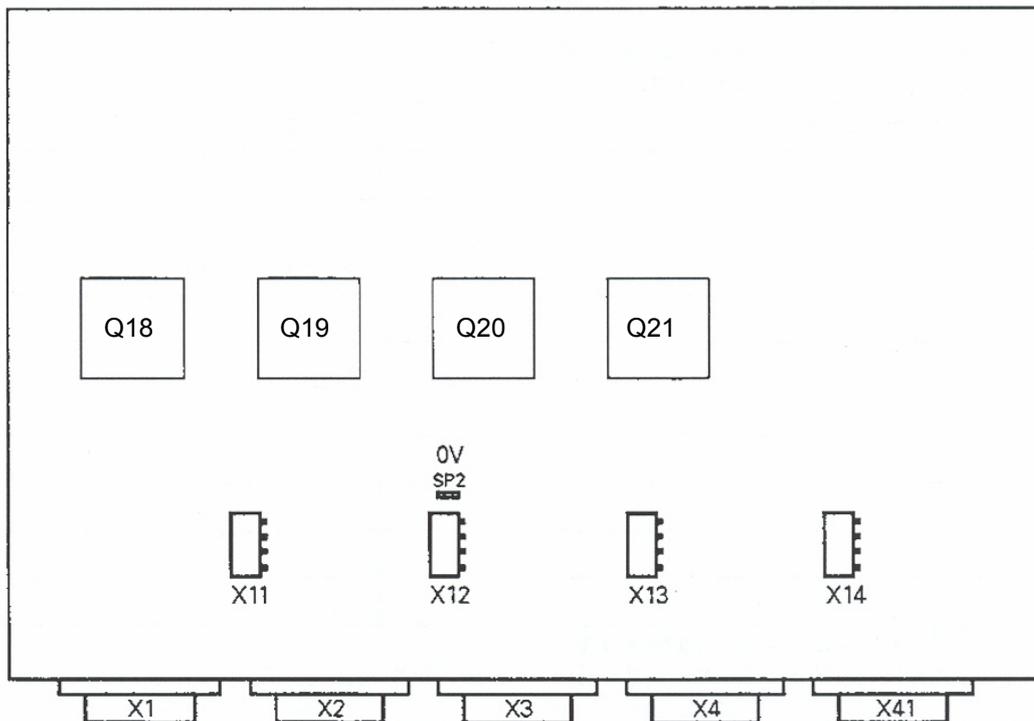


The IK 342 stores the compensation values for phase and amplitude in the electronic potentiometer ICs in non-volatile memory. The offset registers in the counter ICs, however, are volatile (the information is lost when the power is removed). Therefore you have to store the offset compensation values in an EEPROM in the integral component of the electronic potentiometer. After switch-on the offset compensation values must be loaded via software from the EEPROM to the offset registers in the counter ICs. There are two procedures defined for this task in the IIC.CPP file. The "StoreOffset" procedure stores the offset compensation values in the EEPROM and the "LoadOffset" procedure transfers the values from the EEPROM to the offset registers in the counter ICs. 'LoadOffset' is also used by the 'InitIk342' procedure.

The following aids are necessary for the signals to be compensated:

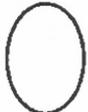
- Oscilloscope for XY-representation
- Digital multimeter
- Software for setting the electronic potentiometer and for describing the offset registers (e.g. POTIS.EXE)

You can calibrate the four encoder input signals at the PCB connectors for encoder outputs designated X11, X12, X13 and X14. There is also a soldering tag on the PCB which is designated 0 V (see diagram).



Compensating phase shift and amplitude ratio

- Connect the oscilloscope as follows:
 - X deflection to Pin 4a of the PCB connector
 - Y deflection to Pin 5a of the PCB connector
 - Grounded at soldering connection designated 0 V
- Move the scanning head, scanning frequency approx. 1 kHz. You can now see a LISSAJOUS figura in the oscilloscope. If both sinusoidal encoder signals are adjusted optimally, you can see a circle. If the signals are not optimally compensated, you can see either a vertically-tendanced or horizontally-tendanced ellipsis. If the phase shift of the two signals is not 90°, then you will see a lopsided ellipsis (see diagram).

Reproduced image	Evaluation
	Amplitudes are the same 90° phase shift Encoder optimally compensated
	Phase shift ≠ 90°
	Amplitudes are not the same

- Compensate the amplitude ratio and the phase angle via software by adjusting the electronic potentiometer.

Compensate offset

- Set the digital multimeter to DC (direct current display) and connect it as follows:
- - For compensating the 0° signal: to Pin 5b (reference potential) and Pin 5a of the PCB connector.
- - For compensating the 90° signal: to Pin 4b (reference potential) and Pin 4a of the PCB connector.
- Read the measured value for constant scanning head traverse. **Pay attention to sign!**
- Determine compensation values for the offset register: 1 step = 4.88 mV. The value to be programmed is calculated as follows:
- Input = - $\frac{\text{scanned mean value mV}}{4.88 \text{ mV}}$

Note that the compensation value must be programmed with the opposite sign to that of the scanned mean value. Do not be surprised if you do not see any change at the measuring point after programming the offset register. The offset register is to be found behind your measuring point, so you will not be able to observe the effect of your compensation value. The effect of the compensation value can only be checked when registers \$10 for the 0° amplitude and \$0C for the 90° amplitude are read.

4.5 External functions

A 9-pin D-sub connector is provided for external latching of measured values and for synchronous latching with several cards. The connector required for this (Id.-Nr. 297 050-ZY) can be ordered from HEIDENHAIN. See "6 Latching position value" for a more detailed description of these functions.

Pin layout of the 9-pin D-sub female connector

Interpretation of signal	Designation of signal	Layout Connection no.
Latch X1, X2, X3, X4 (configurable via latch logic)	E1	3
	E2	4
	E3	5
	E4	6
Cascading 0	-CASC0	7
Cascading 1	-CASC1	8
0V	0V	1
0V	0V	2
+5V	+5V	9

5 Addressing

5.1 Allocation of the address space

The VMEbus counter card occupies 512 bytes of the available address space. The 8-pin S1 DIP switch on the card (with switching levels 1 to 8) divides the A16 address space of the VMEbus (Address Modifier AM: \$29) into 128 sections, each with 512 bytes. Switching level 1 has no function in setting the base address: it is used for the interrupt vector. Only 128 base addresses are possible.

You can calculate the base address of the card as follows:

Base address = switch position (without S1) • 512

Example:

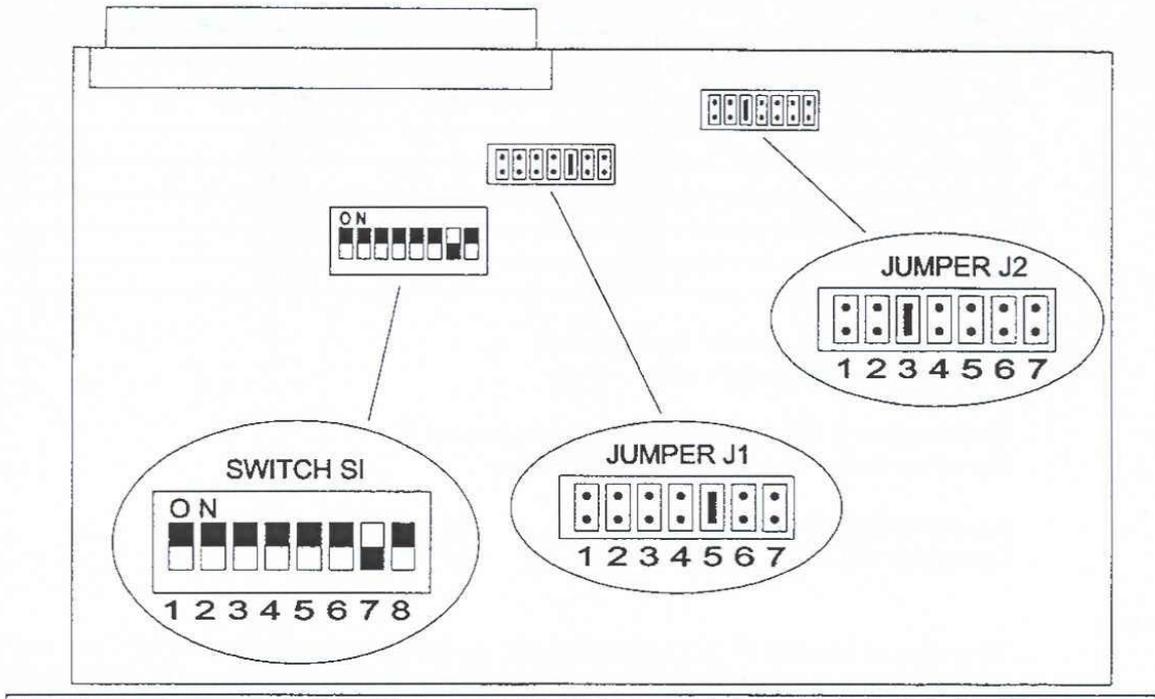
DIP switch S1 = \$A1; base address = (\$A1 shift right 1) • 512 = \$A000

Allocation of the address space:

	Function
BA + \$000	ID register, 1 register (8 bit) read access only
BA + \$080	Counter X1, 32 registers (16 bit)
BA + \$0C0	Counter X2, 32 registers (16 bit)
BA + \$100	Counter X3, 32 registers (16 bit)
BA + \$140	Counter X4, 32 registers (16 bit)
BA + \$180	Latching / configuration logic, 8 registers (8 bit)

Each of the four counters {inputs X1 to X4} has its own register block. The functions of the counter registers and control registers are identical for all four counters. In this description the counter registers and the control registers are given without an offset. If you want to contact one of the registers of counter 4, for example, then you have to add the offset to the register address (base address + \$140).

5.2 Switches and jumpers



Allocation	Jumper 1 Interrupt request line	Jumper 2 Interrupt acknowledge line
1	-IRQ7	-INT1
2	-IRQ6	-INT2
3	-IRQ5	-INT3*
4	-IRQ4	-INT4
5	-IRQ3*	-INT5
6	-IRQ2	-INT6
7	-IRQ1	-INT7

* Factory installed

The interrupt is inhibited if there is no jumper.

DIP switch S1: Base address end IACK vector

Switch no.	Base address	Vector no.
1	0	1
2	1	2
3	2	4
4	4	8
5	8	16
6	16	32
7	32	64
8	64	128

Switch OFF: See table for significance

Switch ON: Significance = 0

Base address = 512 • (sum of the set significances)

Vector number = Sum of the set significances

Factory setting of the base address: 32 = \$20

Factory setting of the vector numbers: 64 = \$40

5.3 Interrupts

The interrupt sources E1 to E4 (external latch inputs) are inhibited after -RESET and can be reenabled if bit 0 of the interrupt enable register of the latching logic is set (see "7.3 Register for configuring the latching logic"). With Jumper J1 you can select the interrupt request line (possible lines: -IRQ7 to -IRQ1) and with Jumper J2 you can select the interrupt acknowledgement line (possible lines: -INT1 to -INT7). An interrupt is stored and output until an interrupt acknowledgement cycle has been successfully completed. If the IK 342 recognizes a valid interrupt acknowledgement cycle, the vector number is sent (DIP switch setting) and the interrupt is cleared.

6 Latching a Position Value

The counter IC block diagram on the last page is useful for the following.

6.1 Overview

In order that the position value can be read, it must first be latched in one of the two data registers 0 or 1.

The following latching methods are available:

- With software: each axis individually or all axes together with internal cascading
- With hardware via X41 (signals E1 to E4)
- With several cards via external cascading: via –CASC0 and -CASC1
- With reference pulse
- With timer

During the latching procedure the IK 342 stores the valid position value completely in one or more data registers without the count procedure being influenced. No other values can then be latched in the registers until the position value has been read.

6.2 Latching with software

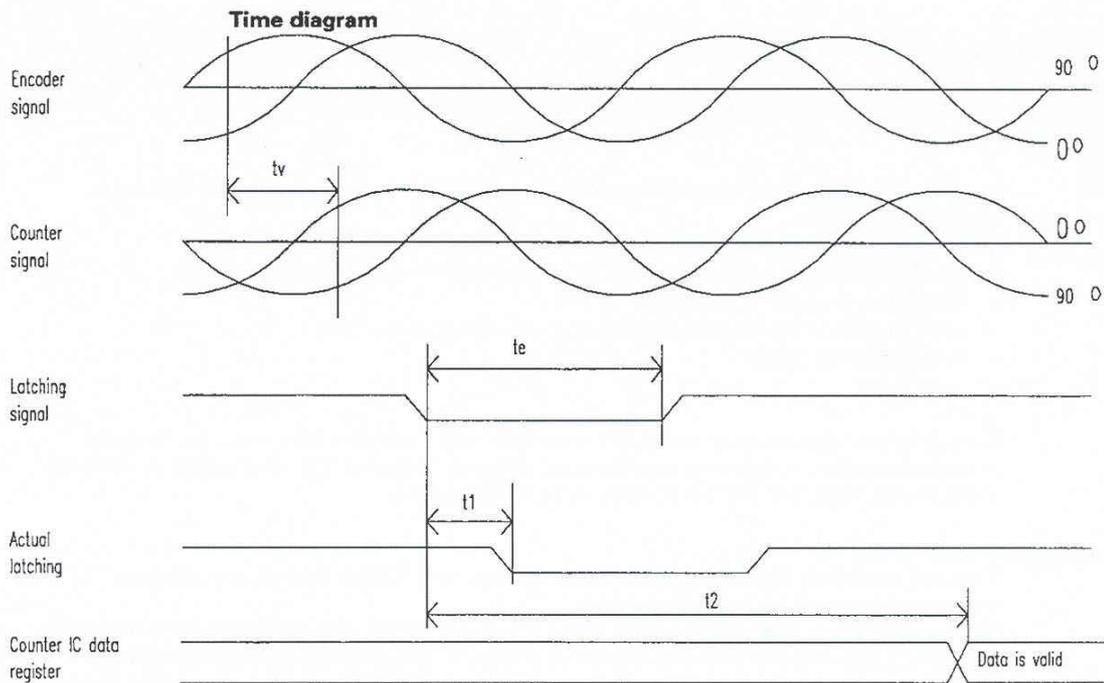
If the corresponding bit of the counter IC control register 1 (\$20) is defined, a latching via software is triggered for the data registers.

Common latching in several counter ICs is carried out via the outputs SYNC0 (data register 0) and SYNC1 (data register 1) of the counter IC of axis X1. The software interrogation via the SYNC lines is enabled by the register for axis cascading (\$18). These signals are passed on by the latching logic to the inputs of further counter ICs via X1.L0 to X4.L0 (SYNC0) and X1.L1 to X4.L1 (SYNC1) and via the enabling register (\$18). In this way the IK 342 can latch all axes at the same time.

In order to have the same propagation time for all axes, you should select a signal path with delay element for axis X1 and without delay element for all other axes.

6.3 Latching with hardware via X41

If E1 to E4 are set to active level, a latching signal is produced in the corresponding data registers of the counter via X1.L0 to X4.L0 or X1.L1 to X4.L1 and the enable register \$18 (see "7.3 Register for configuring the latching logic"). The external latching signals can trigger an interrupt. For this to happen bit 1 must be set at address \$0A in the latching logic, and the interrupt number must be set via jumpers J1 and J2.



Signal	Designation	Min.	Max.
Encoder input lag - counter	t_v	-	< 2 μ s
Width of latch signals E1 to E4	t_e	1.2 μ s	-
latch signals lag	t_1	-	< 0.8 μ s
Lag until the measured value in the data registers is ready to be read	t_2	-	< 24 μ s

Signal levels for X41 (E1 - E4)

Designation	Min.	Max.
U_{eH}	3.5 V	32 V
U_{eL}	-20 V	1 V
I_l	0.5 mA	8 mA

The inputs are low active or high active (configurable via latching logic) and are kept at the corresponding inactive levels by internal pull up /pull down resistances. You can control the inputs with standard TTL, LS, ALS or CMOS components, or with 24 V signals.

6.4 Latching with several cards via external cascading

The X1 axis counter IC controls the two switching outputs $-CASC0$ and $-CASC1$ via the outputs SYNC0 and SYNC1. The two switching outputs $-CASC0$ and $-CASC1$ are connected with the exterior via the 9-pin D-sub connecting element X41. The $-CASC0$ signal corresponds to data register 0 and $-CASC1$ to data register 1.

If the $-CASCx$ outputs are connected externally with the external latch inputs of further cards via X41, then a synchronous latching is possible via several IK 342.

Signals levels for X41 (CASC0, CASC1): TTL

6.5 Latching with reference mark

If the corresponding bits are set in the initialization register (\$24) and the reference mark register (\$1C), the programmed function will be carried out when the reference marks are traversed. In this way you can evaluate both, single and distance-coded, reference marks.

6.6 Latching with timer

You can set a time in the timer register (\$0B), after which an interrogation via the enable register (\$18) will be carried out. The timer is started in the initialization register (\$24).

7 Registers

7.1 BA + \$0000: ID register (read only access)

The JD register contains a hardware identification code for the card.

D8 – D4: PCB version	D3 – D0: Components
0: Version 01	0: IK 342

7.2 Registers 01 the counter ICs

With the IK 342 today the counter IC G38 is used, which replaces the counter IC G26, whereby G38 is upward compatible to G26 and is extended by some read functions to individual registers. The ICs (Q18 to Q21, see picture on page 11) differ not only by their numbers (G26 = 282 150 01, G38 = 315 860 01), but also electrically by the component recognition. (see recognition register of the counter IC).

For the following description, the block diagram of the counter ICs on the last page is helpful.

Register addressing

Access to the counter ICs can be 8-bit or 16-bit (word or byte access). The type of access is programmed in the control register 3 (\$00) bit 0. As the I K 342 has a 16-bit databus, then access to the counter ICs in this case should also be 16-bit. In the following we are assuming 16-bit registers for the register description.

After switch-on the counter ICs are in 8-bit mode (default setting). To revert to 16-bit mode, either bit 0 is set to 0 via word access to address \$00 or byte access to address \$03.

Word access

The address from one word (16-bit) to another is increased by four as line A1 of the VMEbus is connected with A0 of the counter IC. The word addresses are given in the following register overview. An address (e.g. for the initializing register 1 of axis X2) is calculated as follows (with the precondition: DIP switch S1 = \$40):

$$\text{Register address} = ((\$40 \text{ shift right } 1) \cdot \$200) + \$0C0 + \$024 = \$40E4$$

Byte access

For byte access to the high byte, the word addresses of the following register overview are valid. The address of the low byte is calculated as follows:

$$\text{Low byte address} = \text{address from the register overview} + \$03$$

Register overview

Address offset to counter base address	Write access	Read access
\$3C \$38 \$34	No function	Data register 0, LS-Word Data register 0 Data register 0, MS-Word
\$30 \$2C \$28	No function	Data register 1, LS-Word Data register 1 Data register 1, MS-Word
\$24 Low byte High byte	Initialization register 1 Initialization register 2	Initialization register 1 Initialization register 2
\$20 Low byte High byte	Control register 1 No function	Status register 1 Status register 2
\$1C Low byte High byte	Reference mark register No function	No function Amplitude value register
\$18 Low byte High byte	Enable register for interrogation of position value Axis cascading	No function No function
\$14 Low byte High byte	No function No function	No function No function
\$10 Low byte High byte	Offset register for 0° signal No function	Amplitude for 0° signal Amplitude for 0° signal
\$0C Low byte High byte	Offset register for 90° signal No function	Amplitude for 90° signal Amplitude for 90° signal
\$08 Low byte High byte	Timer register, LS-Byte Timer register, MS-Byte	No function No function
\$04 Low byte High byte	Control register 2 No function	Status register 3 Identification register
\$00 Low byte High byte	Control register 3 No function	Status register 4 No function

528 10 S3C: Data registers for the counters

The measured values are stored in 48-bit registers. There are two data registers available for each axis: data register 0 (\$3C to \$34) and data register 1 (\$30 to \$28). The measured values are formed from the 10-bit interpolation value and the 32-bit value of the period counter. This means that only 42 bits of the 48-bit wide register are used for the measured value. The top 6 bits are expanded with the correct sign in two's complement representation.

The 48-bit data width can be narrowed to 32 bits via initialization register 1 (\$24), bit D6.

We can also establish whether the measured value was formed only from the period counter value (data bits D0 to D9 are not defined), or whether the measured value was formed from the period counter value and the interpolation value, using initialization register 1 (\$24), bit D7.

You can store the counter values in the data registers (see also "6 Latching position value") via:

- Software interrogation
- External inputs
- Reference marks
- Timers

You can interrogate whether the measured value has already been latched in the data registers, via the status register 1 (\$20), bit D0 or D1. As long as bit D0 or D1 is set, then no other value can be latched until the uppermost value of the word has been read (exception: with control register 2, bit D6 or D7, latching is enabled without the previous value being read.) In the 48 bit mode these are the data registers \$34h or \$28 and in the 32-bit mode the date registers \$38 or \$2C. After the measured value has been read bit D0 or D1, in status register 1 (\$20), is reset.



If the counter is stopped or if it is stored when the reference marks are traversed, then D0 to D9 will contain the fixed value 256.

\$24: Initialization register 1 (write access)

Bit	Function
D0	Operation with/without interpolation 0 = operation as period counter (without interpolation - data bits 00 to 09 are not defined)
D1	0
D2	Timer 0 = revert timer to 0 and stop 1 = start timer
D3	0
D4	0
D5	0
D6	Latch enable 0 = operation mode: 32 bit register If bits D24 to D31 are read, then the status bit D0 or D1 in the status register 1 (\$20) is reset. 1 = operation mode: 48 bit register If bits D40 to D47 are read, then the status bit D0 or D1 in the status register 1 (\$20) is reset.
D7	Count direction The count direction determines whether the positive traverse direction will count positive or negative (inverse). 0 = normal count direction 1 = inverse count direction The count direction may only be inverse in the period counter mode! With interpolation the inverted count direction would result in an inaccurate gating of interpolation value and period counter value.

\$24: Initialization register 2 (write access)

Bit	Function
D8 D9	<p>Only in period counter operation: Edge evaluation Because of the two incremental encoder signals (0° and 90° el.) there are a maximum of four edges available for evaluation per signal period. The counters can be programmed to count either one, two or four edges per signal period. 00 = 1 edge 01 = 2 edges 11 = 4 edges In operation with an interpolation value. the value for one edge is set automatically.</p>
D10	<p>Only in period counter operation: Counting mode 0 = linear counting mode -2^{41} to $+2^{41} - 1$ 1 = angular counting mode as in D11, for angle encoders with 36 000 or 360 000 lines per revolution.</p>
D11	<p>Only for angle display: Counting mode 0 = 17 999 to -18 000 1 = 179 999 to -180 000</p>
D12	0
D13	0
D14 D15	<p>Measured value interrogation with reference pulse 0 = 1st reference mark stored in data register 0 1 = 1st reference mark stored in data register 1 0 = 2nd reference mark stored in data register 0 1 = 2nd reference mark stored in data register 1</p>

\$24: Read access: Bits D0 to D15: reading back the initialization registers 1 and 2

\$20: Control register 1 (write access)

Bit	Function
D0	1 = software polling: measured value in data register 0
D1	1 = software polling: measured value in data register 1
D2	1 = software polling of all data registers (must be enabled in the interrogation enable register)
D3	1 = start counter
D4	1 = stop counter
D5	1 = cancel counter
D6	1 = clear encoder error (frequency exceeded)
D7	1 = Cancel amplitude register
D8 D9 D10 D11 D12 D13 D14 D15	No function

\$20: Status register 1 (read access)

Bit	Function
D0	Status for software interrogation in data register 0; 1 = value is ready
D1	Status for software interrogation in data register 1; 1 = value is ready
D2 D3	No function
D4	1 = counter has been stopped
D5	Signal amplitude: 1 = signal amplitude ok 0 = signal amplitude too small (defect, contamination)
D6	1 = encoder error (frequency exceeded)
D7	No function

\$20: Status register 2 (read access)

Bit	Function
D8	1 = reference mark approach is active
D9 D10 D11 D12	No function
D13	Logic levels for the 0° signal
D14	Logic levels for the 90° signal
D15	Logic levels for the reference mark

\$1C: Reference marks register (write access)

Bit	Function
D0	1 = start counter
D1	1 = stop counter
D2	1 = cancel counter
D3	1 = interrogate measured value
D4	1 = interrogate measured value as the second reference mark is being traversed
D5	1 = clear counter each time a reference mark is traversed
D6 D7 D8 D9 D10 D11 D12 D13 D14 D15	No function

\$1C: Reference marks/amplitude value register (read access)

Bit	Function															
D0	Status "start counter with REF" (with G38 IC only)															
D1	Status "stop counter with REF" (with G38 IC only)															
D2	Status "clear counter with REF" (with G38 IC only)															
D3	Status "fetch measuring value with REF" (with G38 IC only)															
D4	Status "fetch measuring value by traversing the 2nd reference mark" (with G38 IC only)															
D5	Status "clear counter by traversing each reference mark" (with G38 IC only)															
D6 – D7	No function															
D8 D9	<p>Current amplitude Each position value interrogation gives a new amplitude value.</p> <table border="0"> <tr> <td>D9</td> <td>D8</td> <td>IK342</td> </tr> <tr> <td>0</td> <td>0</td> <td>normal amplitude $0.47 V_{PP} < U_e < 1.41 V_{PP}$</td> </tr> <tr> <td>0</td> <td>1</td> <td>low amplitude $0.22 V_{PP} < U_e < 0.47 V_{PP}$</td> </tr> <tr> <td>1</td> <td>0</td> <td>high amplitude $U_e > 1.41 V_{PP}$</td> </tr> <tr> <td>1</td> <td>1</td> <td>erroneously small amplitude $U_e < 0.22 V_{PP}$</td> </tr> </table> <p>The amplitude value should be frozen in control register 2 before being read via bit D4. The amplitude register is reset by bit D7 in control register 1.</p>	D9	D8	IK342	0	0	normal amplitude $0.47 V_{PP} < U_e < 1.41 V_{PP}$	0	1	low amplitude $0.22 V_{PP} < U_e < 0.47 V_{PP}$	1	0	high amplitude $U_e > 1.41 V_{PP}$	1	1	erroneously small amplitude $U_e < 0.22 V_{PP}$
D9	D8	IK342														
0	0	normal amplitude $0.47 V_{PP} < U_e < 1.41 V_{PP}$														
0	1	low amplitude $0.22 V_{PP} < U_e < 0.47 V_{PP}$														
1	0	high amplitude $U_e > 1.41 V_{PP}$														
1	1	erroneously small amplitude $U_e < 0.22 V_{PP}$														
D10 D11	<p>Minimal amplitude value Coding and read access see bits D8 and D9. If the given amplitude value is lower than the stored minimal value more than four times in a row, then the IK replaces the stored minimal value by the current amplitude value.</p>															
D12	G26: no function															
D13	G38: maximum value of amplitude (as written in "Minimal value of amplitude")															
D14 D15	No function															

\$18: Register for enabling measured value interrogation (write access)

Bit	Function
D0	1 = enable L0 for data register 0
D1	1 = enable L0 via delay circuit (125 ns) for data register 0
D2	1 = enable the “software polling in all data registers” for data register 0
D3	1 = enable the “software interrogation by timer” for data register 0
D4	1 = enable L1 for data register 1
D5	1 = enable L1 via delay circuit (125 ns) for data register 1
D6	1 = enable the “software polling in all data registers” for data register 1
D7	1 = enable the “software interrogation by timer” for data register 1

You can see more clearly how the individual bits function in the counter IC block diagram on the last page of this manual.

\$18: Register for axis cascading (write access)

Bit	Function
D8	1 = enable L0 for all axes (SYNC0)
D9	1 = enable the “software polling in all data registers” for all axes (SYNC0)
D10	1 = enable the timer strobes for all axes (SYNC0)
D11	1 = No function
D12	1 = enable L1 for all axes (SYNC1)
D13	1 = enable the “software polling in all data registers” for all axes (SYNC1)
D14	1 = enable the timer strobes for all axes (SYNC1)
D15	1 = No function

You can see more clearly how the individual bits function in the counter IC block diagram on the last page of this manual.

\$18: Read access:
G26: Register cannot be read back
G38: Complete register can be read back

\$10: Offset register for the 0°-signal (write access)

This register contains the 7.bit offset compensation value for the 0° signal in two's complement representation. The maximum compensation is ± 63 .

The compensation values can only be written if one of the status bits D5 or D6 in status register 3 has the value 0.

Functioning:

Offset compensation values are added to the digital values of the 0° signal (0 to 1023) and 90° signal. In the case of overflow the values are limited to 1023 or 0.

Bit	Function
D0 D1 D2 D3 D4 D5 D6	1 = Offset compensation value for the 0° signal in two's complement representation
D7 D8 D9 D10 D11 D12 D13 D14 D15	1 = No function



In a power failure the contents of the offset register in the counter ICs may be lost. For this reason the offset compensation values are stored in an EEPROM in the electronic potentiometer IC. After switch-on the offset compensation values must be transferred from the EEPROM to the offset registers of the counter ICs (see procedures "StoreOffset" and "LoadOffset").

\$10: Amplitude for the 0° signal (read access)

Each time an analog to digital conversion takes place, the result is stored. Before the values are read they should be frozen via bit D4 in control register 2.

Bit	Function
D0 D1 D2 D3 D4 D5 D6	Amplitude for the 0° signal
D7 D8 D9 D10 D11 D12 D13 D14 D15	No function

\$0C: Offset register for the 90°-signal (write access)

This register contains the 7.bit offset compensation value for the 90° signal. See “16h: Offset register for the 0° signal” for a description of the functioning.

Bit	Function
D0 D1 D2 D3 D4 D5 D6	1 = Offset compensation value for the 90° signal in two's complement representation
D7 D8 D9 D10 D11 D12 D13 D14 D15	1 = No function

\$0C: Amplitude for the 90° signal (read access)

Each time an analog to digital conversion takes place, the result is stored. Before the values are read they should be frozen via bit D4 in control register 2.

Bit	Function
D0 D1 D2 D3 D4 D5 D6	Amplitude for the 90° signal
D7 D8 D9 D10 D11 D12 D13 D14 D15	No function

\$08: Timer register (write access)

The 13-bit timer value (0 to 8191) is stored in the timer register \$08. The cycle time is given in μs whereby 1 μs must be subtracted from the desired cycle time. The clock frequency of the timer is 16 MHz (VMEbus) / 16 = 1 MHz, i.e. an increment of the timer register corresponds to 1 μs .

Example:

Desired cycle time = 1 ms = 1 000 μs

Value programmed = 1000 - 1 = 999

The timer is not yet started when this register is defined. The timer is only started when bit D2 is set in initialization register 1 (\$24). Bit D3 also has to be set in enable register \$18.

Bit	Function
D0	Timer value
D1	
D2	
D3	
D4	
D5	
D6	
D7	
D8	
D9	
D10	
D11	
D12	
D13	No function
D14	
D15	

\$08:**Read access:**

G26: Register cannot be read back

G38: Timer value can be read back

\$04: Control register 2 (write access)

Bit	Function
D0 D1 D2 D3	Program a fixed value ≥ 8
D4	1 = Freeze amplitude for 0° signal (\$10) and 90° signal (\$0C) as well as amplitude register (\$1C high byte). This bit must be set to prevent the register values changing during readout. You can interrogate whether the values have been frozen via bit D4 in status register 3.
D5	0
D6	1 = another interrogation possible via data register 0, without the previous measured value being read. In this mode the value can be changed during readout.
D7	1 = another interrogation possible via data register 1, without the previous measured value being read. In this mode the value can be changed during readout
D8 D9 D10 D11 D12 D13 D14 D15	No function

\$04: Status register 3 (read access)

Bit	Function
D0 D1 D2 D3	G26: Value can not be read back G38: value can be read back
D4	1 = amplitude for 0° signal (\$10) and 90° signal (\$0C) as well as amplitude value register (\$1C high byte) are frozen and can be read.
D5	0 = offset register for 0° signal is written.
D6	0 = offset register for 90° signal is written.
D7	No function

\$04: Identification register (read access)

Bit	Function
D8	IC recognition: G26: fixed value 8 G38: fixed value 9
D9	
D10	
D11	
D12	
D13	
D14	
D15	

\$00: Control register 3 (write access)

Bit	Function
D0	0 = 16-bit bus 1 = 8-bit bus
D1	Fixed value 0
D2	No function
D3	
D4	
D5	
D6	
D7	
D8	
D9	
D10	
D11	
D12	
D13	
D14	
D15	

\$00: Status register 4. (read access)

Bit	Function
D0	Read back D0 (control register 3)
D1	Logic level on pin L0
D2	Logic level on pin L1
D3	No function
D4	
D5	
D6	
D7	
D8	
D9	
D10	
D11	
D12	
D13	
D14	
D15	

7.3 BA + \$180: Registers for configuring the latch logic

The configuring logic consists of six 8-bit registers, of which all can be written and read. After –RESET all registers are 0.

Each time the latch logic is configured, data registers 0 and 1 of the counter ICs have to be read once, as a configuration may trigger latch pulses that inhibit further latch procedures in the counters.

Example

Address for the register for configuration of the inputs E1 to E4

(assuming that DIP switch = \$40):

Register address = ((\$40 shift right 1) • \$200) + \$180 + \$08 = \$4188

The 8-bit register is found in the low byte of this word address.

Overview

Base address configuration logic (BAK): BA + \$180

Address offset	Function
BAK + \$00	Latching configuration X1
BAK + \$02	Latching configuration X2
BAK + \$04	Latching configuration X3
BAK + \$06	Latching configuration X4
BAK + \$08	Configuration of the input levels E1 to E4 Configuration encoder inputs X1, X2
BAK + \$0A	Configuration encoder inputs X2, X3, X4
BAK + \$0C	Latch source
BAK + \$0E	I ² C bus control, interrupt enable/disable (1 bit), excess of encoder input signal level.

\$00 to \$06 Configuration register for the external inputs E1 to E4 (read and write access)

Default value after RESET: 0

Address \$00	Configuration register X1
Address \$02	Configuration register X2
Address \$04	Configuration register X3
Address \$06	Configuration register X4

The bits have the same meaning for every axis:

Bit	Function
D0	0 = E1 is inhibited for this axis 1 = E1 latches this axis
D1	0 = E1 latches data register 0 of this axis 1 = E1 latches data register 1 of this axis
D2	0 = E2 is inhibited for this axis 1 = E2 latches this axis
D3	0 = E2 latches data register 0 of this axis 1 = E2 latches data register 1 of this axis
D4	0 = E3 is inhibited for this axis 1 = E3 latches this axis
D5	0 = E3 latches data register 0 of this axis 1 = E3 latches data register 1 of this axis
D6	0 = E4 is inhibited for this axis 1 = E4 latches this axis
D7	0 = E4 latches data register 0 of this axis 1 = E4 latches data register 1 of this axis

\$08: Configuration register for input levels E1 to E4 and configuration encoder inputs X1, X2 (read and write access)

Default value after RESET: 0

Bit	Function
D0	0 = E1 is low active 1 = E1 is high active
D1	0 = E2 is low active 1 = E2 is high active
D2	0 = E3 is low active 1 = E3 is high active
D3	0 = E4 is low active 1 = E4 is high active
D4	0 = X1: input 1 V_{PP} 1 = X1: input 11 μA_{PP}
D5	Must be set to 1 (absolutely!)
D6	0 = X1: input amplifier fast, 1 V_{PP} : 500 kHz; 11 μA_{PP} : 175 kHz 1 = X1: input amplifier slow (33 kHz)
D7	0 = X2: input 1 V_{PP} 1 = X2: input 11 μA_{PP}

\$0A Configuration register for the encoder inputs X2, X3, X4 (read and write access)
 Default value after RESET: 0

Bit	Function
D0	Must be set to 1 (absolutely!)
D1	0 = X2: input amplifier fast, 1 V _{PP} : 500 kHz; 11 μA _{PP} :175 kHz 1 = X2: input amplifier slow (33 kHz)
D2	0 = X3: input 1 V _{PP} 1 = X3: input 11 μA _{PP}
D3	Must be set to 1 (absolutely!)
D4	0 = X3: input amplifier fast, 1 V _{PP} : 500 kHz; 11 μA _{PP} :175 kHz 1 = X3: input amplifier slow (33 kHz)
D5	0 = X4: input 1 V _{PP} 1 = X4: input 11 μA _{PP}
D6	Must be set to 1 (absolutely!)
D7	0 = X4: input amplifier fast, 1 V _{PP} : 500 kHz; 11 μA _{PP} :175 kHz 1 = X4: input amplifier slow (33 kHz)

\$0C: Register for display of the latch source (read access)
 Default value after RESET: 0

Bit	Function
D0	1 = latch source was E1
D1	1 = latch source was E2
D2	1 = latch source was E3
D3	1 = latch source was E4
D4	1 = latch source was SYNC0
D5	1 = latch source was SYNC1
D6 D7	No function

\$0C: Register for display of the latch source (write access)

Default value after RESET: 0

The accordingly set bit can be reset again with a write access with value "1".

Bit	Function
D0	1 = reset bit 1
D1	1 = reset bit 2
D2	1 = reset bit 3
D3	1 = reset bit 4
D4	1 = reset bit 5
D5	1 = reset bit 6
D6	No function
D7	

\$0E: Configuration register for the I²C bus, interrupt enable/disable, excess of encoder input signal level (read access)

Default value after RESET: 0

The accordingly set bit can be reset again with a write access with value "1".

Bit	Function
D0	pin scan serial clock output I ² C bus
D1	pin scan serial data output I ² C bus
D2	pin scan serial data input I ² C bus
D3	pin scan interrupt disable
D4	0 = excess of encoder input signal level X1 (pin scan)
D5	0 = excess of encoder input signal level X2 (pin scan)
D6	0 = excess of encoder input signal level X3 (pin scan)
D7	0 = excess of encoder input signal level X4 (pin scan)

\$0E: Configuration register for the I²C bus, interrupt enable/disable, excess of encoder input signal level (write access)

Default value after RESET: 0

Bit	Function
D0	0 = set serial clock output I ² C bus 1 = reset serial clock output I ² C bus
D1	0 = set serial data output I ² C bus 1 = reset serial data output I ² C bus
D2	No function
D3	0 = Interrupt disable 1 = Interrupt enable
D4	No function
D5	
D6	
D7	

8. Programming

The programming of an IK 342 is shown in this description using "BORLAND C++" examples. The programs were created and tested on an industrial computer (from ROTEC, D-76411 Rastatt) with an INTEL 486 CPU (DOS version 6.0). VMEbus interface and BORLAND C++ compiler (version 4.5).

The following files can be used for adapting the ISA bus to the VME bus (they are supplied on the accompanying disk):

. VMEROTEC.H and
. VMEINIT.C

We are not going to go into details about the data end functioning of these files here, as this has nothing to do with the functioning of the IK 342.

8.1 Basic functions

You can find the programs described here on the accompanying disk in the subdirectory **SAMPLE1**.

The files

- IK342_0.H and
- IK342_0.C

contain the most important data and function definitions which you will need when working with the IK 342.

The most important functions are listed in the following examples:

Vmelnit()

Initializes the VMEbus. This function is adapted to the industrial computer manufactured by ROTEC. You will have to write your own initialization function for the hardware you use.

WriteRegister

Writes a value to a 16-bit counter IC register.

ReadRegister

Reads a value from a 16-bit counter IC register.

SoftLatch_0 and SoftLatch_1

Stores the current counter value in data register 0 or data register 1.

CountValueLatched

Checks if the measured value was stored.

PollForLatched

Continues the interrogation procedure until a measured value is stored.

ReadCountValue_32

Reads a 32-bit measured value from a counter IC.

ReadCountValue_48

Reads a 48-bit measured value from a counter IC.

SAMPLE32.EXE

The program SAMPLE32.EXE shows a simple application for reading a 32-bit measured value.

Source code: SAMPLE32.C
IK342_0.C
VMEINIT.C
Header files: SAMPLE.H
IK342_0.H
VMEROTEC.H

SAMPLE48.EXE

The program SAMPLE48.EXE shows a simple application for reading a 48 bit measured value.

Source code: SAMPLE48.C
IK342_0.C
VMEINIT.C
Header files: SAMPLE.H
IK342_0.H
VMEROTEC.H

The headerfile IK342_0.H

```
/*-----IK342_0.H-----  
  
DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany  
Header File for the Basic Functions of the IK342  
  
V 1.00  
November 1998  
-----*/  
  
#define CLS printf ("\x1B[2J")  
  
/*-----  
  Defines for register addresses.  
-----*/  
  
#define INITIALIZING REGISTER      0x24  
#define CONTROL REGISTER 1        0x20  
#define CONTROL=REGISTER=2        0x04  
  
//Configuration- and Id-Register  
#define ENLATCHX1  0x00  
#define ENLATCHX2  0x02  
#define ENLATCHX3  0x04  
#define ENLATCHX4  0x06  
#define LATCHPOL   0x08  
#define MSINPUT    0x0A  
#define LATCHSTAT  0x0C  
#define INTREG     0x0E  
  
/*-----  
  Macro to calculate the IK address.  
-----*/  
  
#define CALCULATE_IK_BASE_ADDRESS(switch) \  
  ((unsigned short)(switch >> 1)*0x0200) | 0x8000  
  
/*-----  
  Macro to switch VME to A16 memory space.  
  ROTEC specific code.  
-----*/  
  
#define SWITCH_VME_TO_A16_ADDRESS_SPACE(Switch) \  
  outport (ADR_REG,\  
  (((unsigned short)(switch >> 1)*0x0200)\  
  & 0x8000) >> 8) | 0xFC00)  
  
/*Definitions of functions*/
```

```

void writeRegister (unsigned short, unsigned char,
    unsigned short, unsigned short);

unsigned short ReadRegister (unsigned short, unsigned char,
    unsigned short);

void SoftLatch_0 (unsigned short, unsigned char);

void SoftLatch_1 (unsigned short, unsigned char);

unsigned char CountValueLatched (unsigned short, unsigned char,
    unsigned char);

void PollForLatched (unsigned short, unsigned char,
    unsigned char);

long ReadCountValue_32 (unsigned short, unsigned char,
    unsigned char);

double ReadCountValue_48 (unsigned short, unsigned char,
    unsigned char);

unsigned char ReadConfReg (unsigned short usBaseAddress,
    unsigned short usRegisterAddress);

void WriteConfReg (unsigned short usBaseAddress, unsigned short
usRegisterAddress,
    unsigned char usDatum);

```

The functions in IK342_0.C

```

/*-----IK342_0.C-----*/

DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

Driver Unit for IK342 (Basic Functions)

V 1. 00
November 1998
-----*/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "ik342 0.h"
#include "vmerotec.h"

/*-----Functions-----*/
/*-----
                                WriteRegister
-----
This function writes a value in a 16-bit
register of a counter.
-----Parameters-----
usBaseAddress      base address of the IK 342
ucAxis            axis select - axis      1 to axis 4
usRegisterAddress  address of the register
usDatum           value to write to the register address
usPortAddress      port address in which <usDatum> is written
-----*/

void WriteRegister (unsigned short usBaseAddress, unsigned char ucAxis,
    unsigned short usRegisterAddress,
    unsigned short usDatum)
{
    unsigned short usPortAddress;

    switch (ucAxis)
    {

```

```

        case 1:
        usPortAddress = usBaseAddress + 0x0080 + usRegisterAddress;
        break;
        case 2:
        usPortAddress = usBaseAddress + 0x00C0 + usRegisterAddress;
        break;
        case 3:
        usPortAddress = usBaseAddress + 0x0100 + usRegisterAddress;
        break;
        case 4:
        usPortAddress = usBaseAddress + 0x0140 + usRegisterAddress;
        break;
        default:
        printf ("Wrong axis in function <WriteRegister>");
    }

    /* Write <usDatum> to the counter */
    outpw (usPortAddress, usDatum);
}

/*-----
                                     ReadRegister
-----*/

This function reads a value from a 16-bit
register of a counter.
-----Parameters-----
usBaseAddress      basic address A0 to A9 of the IK 342
ucAxis             axis select - axis 1 or axis 2
usRegisterAddress  address of the register
usPortAddress      port address in which <usDatum> is written
-----*/
unsigned short ReadRegister (unsigned short usBaseAddress,
                            unsigned char ucAxis, unsigned short usRegisterAddress)
{
    unsigned short usPortAddress;

    /* Calculate port usRegisterAddress */
    switch (ucAxis)
    {
        case 1:
        usPortAddress = usBaseAddress + 0x0080 + usRegisterAddress;
        break;
        case 2:
        usPortAddress = usBaseAddress + 0x00C0 + usRegisterAddress;
        break;
        case 3:
        usPortAddress = usBaseAddress + 0x0100 + usRegisterAddress;
        break;
        case 4:
        usPortAddress = usBaseAddress + 0x0140 + usRegisterAddress;
        break;
        default:
        printf ("Wrong axis in function <writeRegister>");
    }

    /* Read <usDatum> from the counter */
    return (inpw(usPortAddress));
}

/*-----
                                     SoftLatch_0
-----*/

This function reads the measured value and stores
it in data register 0.
-----*/
void SoftLatch_0 (unsigned short usBaseAddress, unsigned char ucAxis)
{
    WriteRegister (usBaseAddress, ucAxis, 0x20, 0x0001);
}

```

```

/*-----
                                     SoftLatch_1
-----
This function reads the measured value and stores
it in data register 1.
-----*/
void SoftLatch_1 (unsigned short usBaseAddress, unsigned char ucAxis)
{
WriteRegister (usBaseAddress, ucAxis, 0x20, 0x0002);
}

/*-----
                                     CountValueLatched
-----
This function checks whether a measured value is
latched in data register 0 or 1.
-----*/

unsigned char CountValueLatched (unsigned short usBaseAddress,
                                unsigned char ucAxis, unsigned char ucRegister)
{
unsigned char result;
switch (ucRegister)
    {
    case 0:
        result = (unsigned char)
            (ReadRegister (usBaseAddress, ucAxis, 0x020) & 0x0001);
        break;
    case 1:
        result = (unsigned char)
            (ReadRegister (usBaseAddress, ucAxis, 0x020) & 0x0002);
        break;
    }
return (result);
}

/*-----
                                     PollForLatched
-----
This function polls until a measured value is
latched in data register 0 or 1.
-----*/

void PollForLatched (unsigned short usBaseAddress,
                    unsigned char ucAxis,
                    unsigned char ucRegister)
{
switch (ucRegister)
    {
    case 0:
        while (CountValueLatched (usBaseAddress, ucAxis, 0) == 0)
            ;
        break;

    case 1:
        while (CountValueLatched (usBaseAddress, ucAxis, 1) == 0)
            ;
        break;
    }
}

/*-----
                                     ReadCount_Value_32
-----
This function reads 32 bits of a measured value.
-----*/

```

```

long ReadCountValue_32 (unsigned short usBaseAddress,
                        unsigned char ucAxis, unsigned char ucRegister)
{
union    mapper
{
    {
        long field0;
        unsigned short field1[2];
    }buffer;
switch (ucRegister)
{
    case 0:
        buffer.field1[0] = ReadRegister (usBaseAddress, ucAxis, 0x3c);
        buffer.field1[1] = ReadRegister (usBaseAddress, ucAxis, 0x38);
        break;
    case 1:
        buffer.field1[0] = ReadRegister (usBaseAddress, ucAxis, 0x30);
        buffer.field1[1] = ReadRegister (usBaseAddress, ucAxis, 0x2C);
        break;
}
return (buffer.field0);
}

/*-----
                                ReadCouD_Value_48
-----
This function reads 48 bits of a measured value.
-----*/
double ReadCountValue_48 (unsigned short usBaseAddress,
                          unsigned char ucAxis, unsigned char ucRegister)
{
unsigned short usField[3];
double count_value48;

switch (ucRegister)
{
    case 0:
        usField[0] = ReadRegister (usBaseAddress, ucAxis, 0x3C);
        usField[1] = ReadRegister (usBaseAddress, ucAxis, 0x38);
        usField[2] = ReadRegister (usBaseAddress, ucAxis, 0x34);
        break;
    case 1:
        usField[0] = ReadRegister (usBaseAddress, ucAxis, 0x30);
        usField[1] = ReadRegister (usBaseAddress, ucAxis, 0x2C);
        usField[2] = ReadRegister (usBaseAddress, ucAxis, 0x28);
        break;
}

if (usField[2] & 0x8000)
    count_value48 = (double)((usField[0] 65535.0 +
        65536.0*(usField[1]-65535.0)+
        4294967296.0*(usField[2]-65535.0)-1);
    else
        count value48 = (double)(usField[0] +
        65536.0*usField[1] +
        4294967296.0*usField[2];

return (count value48);
}

/*-----
                                ReadConfReg
-----
This function reads one configuration register from IK342
-----Parameters-----
-----*/
unsigned char ReadConfReg(unsigned short usBaseAddress,

```

```

        unsigned short usRegisterAddress)
{
    unsigned short usPortAddress;
    usPortAddress = usBaseAddress + 0x0180 + usRegisterAddress;
    return (inp(usPortAddress+1));
}

/*-----
                                WriteConfReg
-----*/
This function writes the config-register from IK342
-----Parameters-----
-----*/
void WriteConfReg(unsigned short usBaseAddress, unsigned short usRegisterAddress,
                  unsigned char usDatum)
{
    unsigned short usPortAddress;

    usPortAddress = usBaseAddress + 0x0180 + usRegisterAddress;
    outp(usPortAddress+1, usDatum);    // write <usDatum> to the IK342
}

```

The header-file SAMPLE.H

```

/*-----SAMPLE.H-----
DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

Header File for SAMPLE32.C and SAMPLE48.C of the IK342 examples

V 1.00
xxxx 199x-----*/
-----*/

/*-----
Setting of the DIP switch on board of the IK 342
-----*/

#define DIP_SWITCH    0x40

```

The program example SAMPLE32.C

```

/*-----SAMPLE32.C-----
DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany

A simple program for the IK 342 to display
four axes. Measured value with 32 bits.

V 1.00
November 1998

Project files:    IK342_0.C, SAMPLE32.C, VMEINIT.C
Include files:   IK342_0.H, SAMPLE.H, VMEROTEC.H
-----*/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "ik342_0.h"
#include "sample.h"
#include "vmerotec.h"

int main()
{
    double dCountValue1, dCountvalue2, dCountValue3, dCountValue4;

```

```

unsigned short usBaseAddress;

CLS;

/*Initialize VME interface (ROTEC specific functions)*/
VmeInit();
gotoxy(1, 20);
puts("VMEbus initialized. Press any key!");
getch();
CLS;

usBaseAddress = CALCULATE_IK_BASE_ADDRESS(DIP_SWITCH);
SWITCH_VME_TO_A16_ADDRESS_SPACE(DIP_SWITCH);

/* Set 1Vss 500kHz */

WriteConfReg (usBaseAddress, LATCHPOL, 0x20);
WriteConfReg (usBaseAddress, MSINPUT, 0x49);

/* Initialize the board in interpolation mode,
axis 1 */
WriteRegister (usBaseAddress, 1, INITIALIZING_REGISTER, 0x0001);
/* Initialize the board in interpolation mode,
axis 2 */
WriteRegister (usBaseAddress, 2, INITIALIZING_REGISTER, 0x0001);
/* Initialize the board in interpolation mode,
axis 3 */
WriteRegister (usBaseAddress, 3, INITIALIZING_REGISTER, 0x0001);
/* Initialize the board in interpolation mode,
axis 4 */
WriteRegister (usBaseAddress, 4, INITIALIZING_REGISTER, 0x0001);

/* Reset error bit, start counter, axis 1 */
WriteRegister (usBaseAddress, 1, CONTROL_REGISTER_1, 0x0048);
/* Reset error bit, start counter, axis 2 */
WriteRegister (usBaseAddress, 2, CONTROL_REGISTER_1, 0x0048);
/* Reset error bit, start counter, axis 3 */
WriteRegister (usBaseAddress, 3, CONTROL_REGISTER_1, 0x0048);
/* Reset error bit, start counter, axis 4 */
WriteRegister (usBaseAddress, 4, CONTROL_REGISTER_1, 0x0048);

/* Write to control register 2, axis 1 */
WriteRegister (usBaseAddress, 1, CONTROL_REGISTER_2, 0x0008);
/* Write to control register 2, axis 2 */
WriteRegister (usBaseAddress, 2, CONTROL_REGISTER_2, 0x0008);
/* Write to control register 2, axis 3 */
WriteRegister (usBaseAddress, 3, CONTROL_REGISTER_2, 0x0008);
/* Write to control register 2, axis 4 */
WriteRegister (usBaseAddress, 4, CONTROL_REGISTER_2, 0x0008);

/*Cursor off*/
_setcursortype(_NOCURSOR);

while(!kbhit())
{
    /* Software latch in register 0, axis 1 */
    SoftLatch 0 (usBaseAddress, 1);
    /* Software latch in register 0, axis 2 */
    SoftLatch_0 (usBaseAddress, 2);
    /* Software latch in register 0, axis 3 */
    SoftLatch_0 (usBaseAddress, 3);
    /* Software latch in register 0, axis 4 */
    SoftLatch 0 (usBaseAddress, 4);
    /* Poll whether latched in axis 1 */
    PollForLatched (usBaseAddress, 1, 0);
    /* Read axis 1 */

```

```

    dCountValue1 = (double)ReadCountValue_32 (usBaseAddress, 1, 0);
                /* Poll whether latched in axis 2 */
    pollForLatched (usBaseAddress, 2, 0);
                /* Read axis 2 */
    dCountValue2 = (double)ReadCountvalue_32 (usBaseAddress, 2, 0);
                /* PoIl whether latched in axis 3 */
    PollForLatched (usBaseAddress, 3, 0);
                /* Read axis 3 */
    dCountValue3 = (double)ReadCountValue_32 (usBaseAddress, 3, 0);
                /* Poll whether latched in axis 4 */
    PollForLatched (usBaseAddress, 4, 0);
                /* Read axis 4 */
    dCountValue4 = (double)ReadCountValue_32 (usBaseAddress, 4, 0);
                /* Display measured values */
    gotoxy(5,5);
    printf("%16.4f\t%16.4f", dCountValue1*0.02/1024,
                dCountValue2*0.02/1024);

    gotoxy(5,10);
    printf("%16.4f\t%16.4f", dCountValue3*0.02/1024,
                dCountValue4*0.02/1024);

    }
    /*Cursor on*/
    _setcursortype (_NORMALCURSOR);

return (0);
}

```

The program example SAMPLE48.C

```

/*-----SAMPLE48.C-----*/

DR. JOHANNES HEIDENHAIN GmbH, Traunreut, Germany
A simple program for the IK 342 to display
four axes. Measured value with 48 bits.

V 1. 00
November 1998

Project files:          IK342_O.C, SAMPLE48.C, VMEINIT.C
Include files:         IK342_O.H, SAMPLE.H, VMEROTEC.H
-----*/

#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include "ik342 O.h"
#include "sample.h"
#include"vmerotec.h"

int main()
{
double dCountValue1, dCountValue2, dCountValue3, dCountValue4;
unsigned short usBaseAddress;

CLS;

    /*Initialize VME interface (ROTEC specific functions)*/
VmeInit();
gotoxy(1, 20);
puts("VMEbus initialized. Press any key!");
getch();
CLS;

usBaseAddress = CALCULATE_IK_BASE_ADDRESS(DIP_SWITCH);
SWITCH_VME_TO_AI6_ADDRESS_SPACE(DIP_SWITCH);

```

```

    /* Set 1Vss 500kHz, all axes */
WriteConfReg (usBaseAddress, LATCHPOL, 0x20);
WriteConfReg (usBaseAddress, MSINPUT, 0x49);

    /* Initialise the board in interpolation mode,
axis 1 */
writeRegister (usBaseAddress, 1, INITIALIZING_REGISTER, 0x0041);
    /* Initialise the board in interpolation mode,
axis 2 */
WriteRegister (usBaseAddress, 2, INITIALIZING_REGISTER, 0x0041);
    /* Initialise the board in interpolation mode,
axis 3 */
WriteRegister (usBaseAddress, 3, INITIALIZING_REGISTER, 0x0041);
    /* Initialise the board in interpolation mode,
axis 4 */
writeRegister (usBaseAddress, 4, INITIALIZING_REGISTER, 0x0041);

    /* Reset error bit, start counter, axis 1 */
WriteRegister (usBaseAddress, 1, CONTROL_REGISTER_1, 0x0048);
    /* Reset error bit, start counter, axis 2 */
WriteRegister (usBaseAddress, 2, CONTROL_REGISTER_1, 0x0048);
    /* Reset error bit, start counter, axis 3 */
WriteRegister (usBaseAddress, 3, CONTROL_REGISTER_1, 0x0048);
    /* Reset error bit, start counter, axis 4 */
WriteRegister (usBaseAddress, 4, CONTROL_REGISTER_1, 0x0048);

    /* Write to control register 2, axis 1 */
WriteRegister (usBaseAddress, 1, CONTROL_REGISTER_2, 0x0008);
    /* Write to control register 2, axis 2 */
WriteRegister (usBaseAddress, 2, CONTROL_REGISTER_2, 0x0008);
    /* Write to control register 2, axis 3 */
WriteRegister (usBaseAddress, 3, CONTROL_REGISTER_2, 0x0008);
    /* Write to control register 2, axis 4 */
WriteRegister (usBaseAddress, 4, CONTROL_REGISTER_2, 0x0008);

/*Cursor off*/
_setcursortype(_NOCURS);

while(!kbhit())
{
    /* Software latch in register 0, axis 1 */
SoftLatch_O (usBaseAddress, 1);
    /* Software latch in register 0, axis 2 */
SoftLatch_O (usBaseAddress, 2);
    /* Software latch in register 0, axis 3 */
SoftLatch_O (usBaseAddress, 3);
    /* Software latch in register 0, axis 4 */
SoftLatch_O (usBaseAddress, 4);

    /* Poll whether latched in axis 1 */
PollForLatched (usBaseAddress, 1, 0);
    /* Read axis 1 */
dCountValue1 = ReadCountValue_48 (usBaseAddress, 1, 0);
    /* Poll whether latched in axis 2 */
PollForLatched (usBaseAddress, 2, 0);
    /* Read axis 2 */
dCountValue2 = ReadCountValue_48 (usBaseAddress, 2, 0);
    /* Poll whether latched in axis 3 */
PollForLatched (usBaseAddress, 3, 0);
    /* Read axis 3 */
dCountValue3 = ReadCountValue_48 (usBaseAddress, 3, 0);
    /* Poll whether latched in axis 4 */
PollForLatched (usBaseAddress, 4, 0);
    /* Read axis 4 */
dCountValue4 = ReadCountValue_48 (usBaseAddress, 4, 0);

    /* Display measured values */

```

```

gotoxy(5,5);
printf("%16.4f\t%16.4f", dCountValue1*0.02/1024,
      dCountValue2*0.02/1024);

gotoxy(5,10);
printf("%16.4f\t%16.4f", dCountValue3*0.02/1024,
      dCountValue4*0.02/1024);

}
/*Cursor on*/
_setcursortype (_NORMALCURSOR);

return (0);
}

```

8.2 Functions for RAM storage model

You can find examples with the RAM model in the **SAMPLE2** sub-directory.

The data structures and functions used are defined and explained in the following files:

IK342.H:	In this header file you can set the addresses for up to 16 IK 342.
IK342_1.H:	Data structures for a RAM storage model for the I K 342 registers and explanations of the functions in the files IK342_1.CPP, IIC.CPP and POTI_1.CPP.
IK342_1.CPP:	Basic functions for the IK 342.
IIC.CPP:	Functions for data transfer via I ² C-bus.
POTI_1.CPP:	Functions for setting the electronic potentiometer.

In the file IK342_1.H a RAM storage model for the IK 342 registers is set up with the help of data structures. The data for the RAM model is written to the IK 342 registers with the help of the procedures **InitHandler** and **CommHandler**.

POTIS.EXE

The program POTIS.EXE shows how you can set the electronic potentiometer of the IK 342 via the I²C-bus using software.

Source code: POTIS.CPP
IK342_1.CPP
IIC.CPP
POTI1.CPP
VMEINIT.C

Header files: IK342.H
IK342_1.H
VMEROTEC.H

DISPLAY.EXE

The program DISPLAY.EXE shows the contents of the IK 342 data registers.

Source code: DISPLAY.CPP
IK342_1.CPP
IIC.CPP
VMEINIT.C

Header files: IK342.H
IK342_1.H
VMEROTEC.H

10 Block Diagram of the Latch Paths in the Counter ICs

The following block diagram shows:

- How the latch signals affect the data registers
- The function of the individual bits of the enable register for latching measured values
- Axis cascading and the relevant bits from the register with the same name
- The register for I²C-bus control

The delay circuits with a delay time of 125 ns are used in synchronized latching of several axes to compensate the propagation time between the axes. In synchronized latching you should select a signal path with delay circuit for axis 1 and a signal path without delay circuit for all other axes. As the same counter ICs are used for all axes, delay circuits exist in all axes. It is important to note that not all signal path combinations are useful!

Functions of the latching logic

Simultaneous latching of data registers over several axes

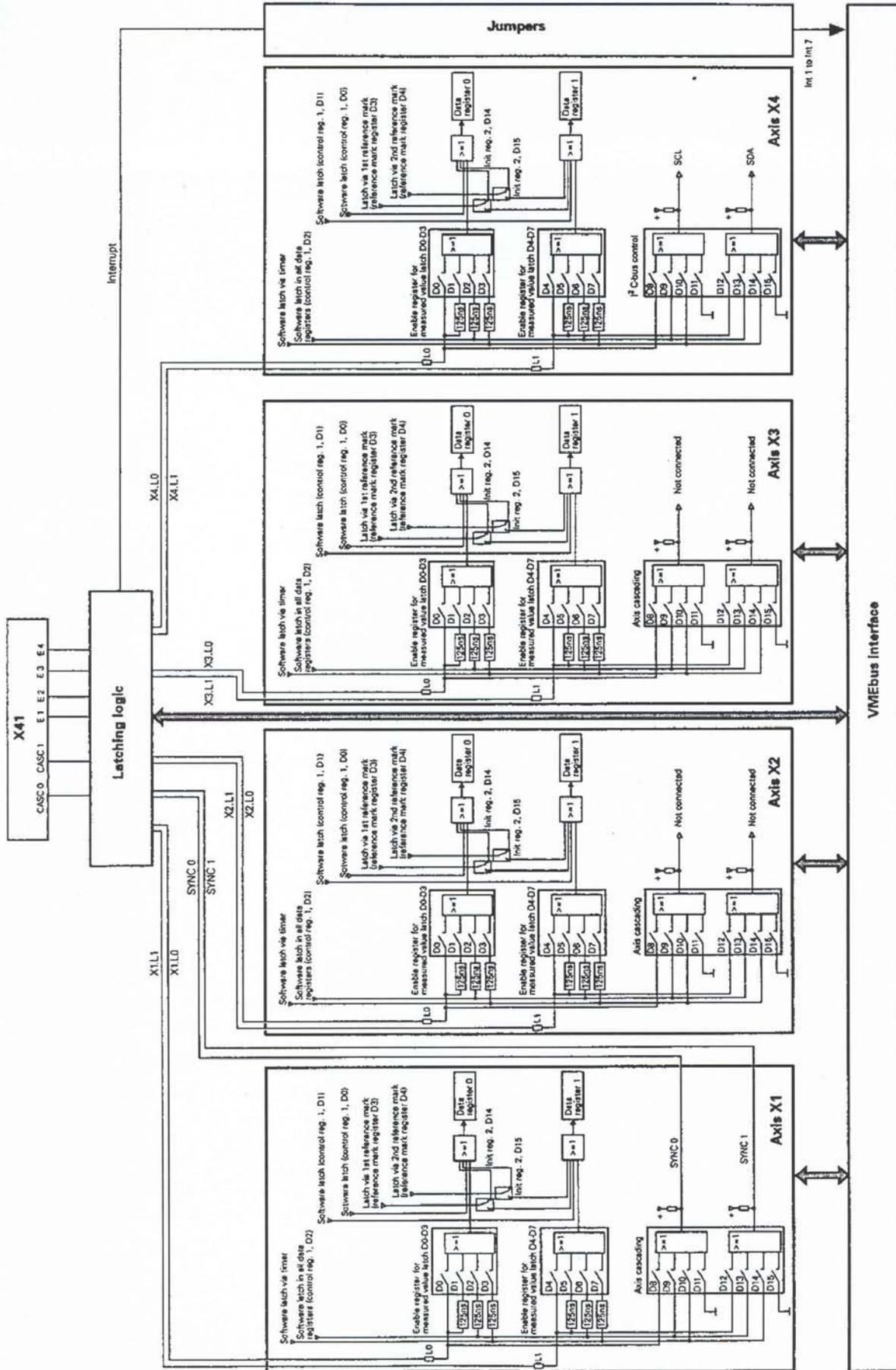
The output SYNC0 of the counter IC for axis 1 is connected with the inputs X1.L0 to X4.L0 via latching logic, and the output SYNC1 with the inputs X1.L1 to X4.L1 (internally wired). Therefore it is possible to latch all axes at the same time.

Simultaneous latching with several cards

The output SYNC0 of the counter IC for axis 1 is connected with output CASC0 via the latching logic to connector X41, and the output SYNC1 is connected with output CASC1 (internally wired). Therefore it is possible to latch the counter ICs of several IK 342 cards at the same time.

Configurable inputs

The inputs E1 to E4 are assigned to the latch inputs X1.L0 to X4.L0 and X1.L1 to X4.L1 via the latching logic (can be programmed via register). Moreover it can also be established whether a signal to the inputs E1 to E4 should generate an interrupt.



HEIDENHAIN

DR. JOHANNES HEIDENHAIN GmbH

Dr.-Johannes-Heidenhain-Straße 5

83301 Traunreut, Germany

☎ +49/86 69/31-0

☎ +49/86 69/50 61

e-mail: info@heidenhain.de

Technical support ☎ +49/86 69/31-10 00

Measuring systems ☎ +49/86 69/31-31 04

e-mail: service.ms-support@heidenhain.de

TNC support ☎ +49/86 69/31-31 01

e-mail: service.nc-support@heidenhain.de

NC programming ☎ +49/86 69/31-31 03

e-mail: service.nc-pgm@heidenhain.de

PLC programming ☎ +49/86 69/31-31 02

e-mail: service.plc@heidenhain.de

Lathe controls ☎ +49/7 11 95 28 03-0

e-mail: service.hsf@heidenhain.de

www.heidenhain.de